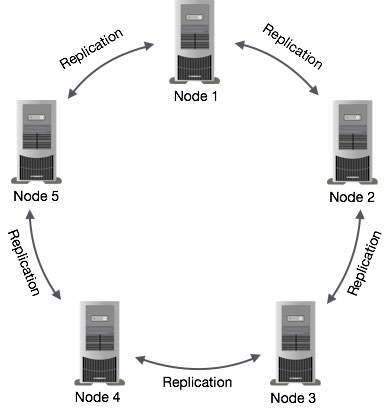
1. Cassandra architecture

Distributed features of Dynamo

Data model and storage from BigTable

* All nodes are independent
* Any node can accept read and write requests
* Other nodes serve, when a node is down

Trong Cassandra, một hoặc nhiều node trong một cluster sẽ đóng vai trò là những bản sao của cùng một dữ liệu. Nếu dữ liệu được xác định là đã quá hạn, Cassandra tự động trả về giá trị dữ liệu gần nhất tới người dùng. Sau đó, Cassandra sẽ tiến hành cập nhật (read repair) ở nững node 

chứa dữ liệu lỗi thời.

Components:

* Node: Nơi chứa dữ liệu
* Data center: Danh sách các node liên quan tới nhau
* Cluster: Chứa 1 hoặc nhiều data center
* Commit log: Là cơ chế khôi phục sau crash trong Cassandra. Mỗi một hành động ghi đều được ghi nhận trong commit log.
* Memtable: Là cấu trúc dữ liệu trong bộ nhớ Sau khi ghi nhận trong commit log, dú liệu được ghi nhận trong mem-table. Đôi khi đối với một gia đình cột, sẽ có nhiều mem-table.

+ is basically a write-back cache of data rows that can be looked up by key -- that is, unlike a write-through cache, writes are batched up in the Memtable until it is full, when it is flushed.

+ <https://stackoverflow.com/questions/27087912/write-back-vs-write-through>

* SSTable: Là một file trên disk, dữu liệu sẽ được đổ về từ mem-table khi mức dữ liệu ở đây chạm tới ngưỡng.
* Bloom filter:

1. Data model

Một data model trong Cassandra khác biệt đáng kể so với những khái niệm của một RDBMS thông thường.

1. Cơ chế:

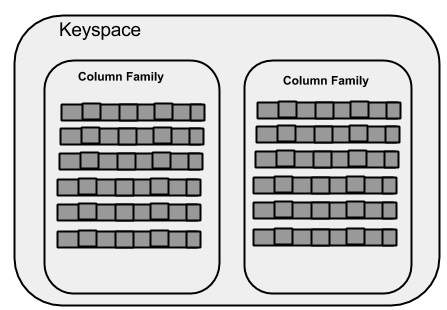
+ Cluster:

CSDL Cassandra được phân phối trên nhiều máy chủ vận hành cunngf nhau. Lớp chứa bên ngoài nhất được gọi là Cluster. Để phòng tránh các rủi ro, mỗi một node bao gồm 1 bản thay thế, khi có lỗi xảy ra, bản thay thế sẽ được gọi thay cho bản gốc. Cassandra sắp xếp các node này trong 1 cluster, theo ring format, và phân dữ liệu cho chúng.

+ Key space:

Key space là lớp container ngoài nhất để chứa dữ liệu trong Cassandra. Các đặc trưng cơ bản của Keyspace là:

* Yếu tố sao chép: Số lượng máy chủ trong mạng cluster nhận các bản sao của cùng một dữ liệu
* Chiến lược thay thế bản sao: Gồm các chiến lược như “**simple strategy**” (rack-aware strategy), “**old network topology strategy**” (rack-aware strategy), và “**network topology strategy**” (datacenter-shared strategy).
* Các column families: Keyspace chứa danh sách các gia đình cột. Một gia đình cột, lại chứa các danh sách các hàng. Một hàng lại chứa các cột đã được sắp xếp. Các gia đình cột tượng trưng cho cấu trúc của dữ liệu. Mỗi keyspace bao gồm ít nhất 1 hoặc nhiều gia đình cột.



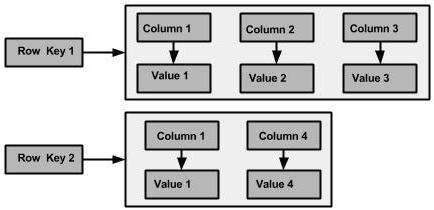
+ Column family:

Chứ một danh sách các dòng đã được sắp xếp. Mỗi dòng, lại chứa các cột theo thứ tự. Bảng dưới đây liệt kê sự khác biệt giữa các gia đình cột và một bảng của các hệ CSDL quan hệ:

|  |  |
| --- | --- |
| **Bảng theo mô hình dữ liệu quan hệ** | **Gia đình cột trong Cassandra** |
| Schema trong mô hình dữ liệu quan hệ là không đổi. Một khi người dùng khai báo một số lượng cột nhất định cho một bảng, tất cả các dòng và các cột phải có dữ liệu (nếu dữ liệu trống phải để là null) | Trong Cassandra, mặc dù các gia đình cột đều được khai báo trước, người dùng vẫn có thể tự do thêm cột vào bất kỳ một gia đình cột nào. |
| Các bảng theo mô hình dữ liệu quan hệ chỉ khai báo các cột và người dùng sẽ điền giá trị vào bảng. | Trong Cassandra, một bảng gồm nhiều cột, hoặc được khai báo nhu một siêu gia đình cột. |

Các đặc trưng cơ bản của một column family trong Cassandra:

* **keys\_cached:** Tượng trưng cho số lượng khu vực lưu cache mỗi SSTable
* **rows\_cached:** Tượng trưng cho số lượng dòng được cached trong bộ nhớ
* **preload\_row\_cache:** It specifies whether you want to pre-populate the row cache.



Column family trong Cassandra

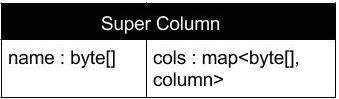
+ Column

* Cấu trúc dữ liệu cơ bản của Cassandra với 3 giá trị, gồm khóa (tên cột), giá trị và time stamp. Bên dưới là cấu trúc của một cột



+ Super Column

* Là cột đặc biệt, là 1 cặp key-value, lưu trữ các sub-column.
* Thông thường, các column families được lưu trữ trên đĩa trong các tập tin riêng biệt. Do vậy, để cải thiện tốc độ của chương trình, người dùng nên lưu trữ các column thường truy vấn trong cùng một column family, ở đây chính là super column. Cấu trúc của một super column:

  
Mô hình dữ liệu của Cassandra và RDBMS:

|  |  |
| --- | --- |
| **RDBMS** | **Cassandra** |
| Xử lý các dữ liệu có cấu trúc | Xử lý dữ liệu không cấu trúc |
| Có schema cố định | Có schema linh hoạt |
| Bảng là một mảng của các mảng (dòng x cột) | Một bảng là danh sách các cặp key-value (ROW x COLUMN key x COLUMN value) |
| CSDL (Database) là đơn vị chứa dữ liệu lớn nhất của mỗi ứng dụng | Keyspace là đơn vị chứa dữ liệu lớn nhất |
| Bảng là các thực thể của Database | Bảng hoặc các họ cột là những thực thể của một keyspace. |
| Dòng là một đơn vị dữ liệu trong CSDL quan hệ | Dòng là một đơn vị bản sao trong Cassandra. |
| Cột là thuộc tính của một quan hệ | Cột là đơn vị lưu trữ trong Cassandra |
| Có khái niệm khóa ngoại, kết bảng. | Các mối liên hệ được biểu hiện bằng cách sử dụng các Collections. |

Nguồn: <https://www.tutorialspoint.com/cassandra/cassandra_data_model.htm>

1. Một số nguyên tắc cơ bản khi thiết kế một data model
2. Storage model

* The hinted handoff feature plus Cassandra conformance and non-conformance to the ACID (atomic, consistent, isolated, durable) database properties are key concepts to understand reads and writes

Không giống với RDBMS, Cassandra ko hỗ trợ đầy đủ các tính năng trong ACID không có locking hay độc lập giao tác khi update nhiều dòng hoặc column family. ACID là từ viết tắt dùng để miêu tả 4 hành vi mà transaction phải đạt được trong RDBMS:

* Atomic: Mọi thứ trong transaction đó phải thành công hoặc tất cả phải đc roll back.
* Consistent: Một transaction không thể để DB ở trạng tháng thiếu nhất quán.
* Isolated: Transaction này phải độc lập với những transaction khác.
* Durable: Phải bền bỉ với các hiện tượng hư hỏng của server.
* In Cassandra, consistency refers to how up-to-date and synchronized a row of data is on all of its replicas.

How is data written?

1. Logging data in the commit log
2. Writing data to the memtable
3. Flushing data from the memtable

Flushing

The process of turning a Memtable into a SSTable is called flushing. You can manually trigger flush via jmx (e.g. with bin/nodetool), which you may want to do before restarting nodes since it will reduce CommitLog replay time. Memtables are sorted by key and then written out sequentially. Thus, writes are extremely fast, costing only a commitlog append and an amortized sequential write for the flush.

Once flushed, SSTable files are immutable; no further writes may be done. So, on the read path, the server must (potentially, although it uses tricks like bloom filters to avoid doing so unnecessarily) combine row fragments from all the SSTables on disk, as well as any unflushed Memtables, to produce the requested data.

1. Storing data on disk in SSTables

Cassandra viết lần đầu tiên vào commit log (đảm bảo durability). Và sau đó là viết vào cấu trúc bảng trong bộ nhớ gọi là memtable. Tác vụ viết là thành công khi nó viết vào commit log và vào memory. Vì vậy có rất ít tác vụ đĩa tại thời điểm đọc ghi dữ liệu. Việc ghi được thực hiện định kì vào đĩa cho một cấu trúc bảng nhất quán gọi là SSTable. Memtable và SSTable được bảo trì theo column family. Memtable được tổ chức bằng việc sắp xếp theo row key và được đẩy xuống SSTable một cách tuần tự.

SSTable là inmutable (không thay đổi). Chúng không được viết lại sau khi đã flush. Có nghĩa rằng một row được lưu trữ thông qua nhiều file trong SSTable. Tại thời điểm đọc, Một dòng phải được combine từ tất cả các SSTable trên đĩa để sinh ra dữ liệu được yêu cầu. Để tối ưu process này Cassandra đãdùng một cấu trúc trong bộ nhớ gọi là bloom filter. Mỗi SSTable có một bloom filter kết hợp với nó. Nó dùng để kiểm tra nếu một row key được yêu cầu tồn tại trong SSTable trước khi làm tác vụ tìm kiếm trên đĩa.